WHAT IS CLAIMED IS:

1. A method for the automatic distribution, review and revocation of user and group permissions to objects through management of role permissions to abstract objects in a computing environment comprises a role-based access control system that includes a directed acyclic graph representing role-membership inheritance relationships and a directed acyclic graph representing role-permission inheritance relationships, said method comprising:

associating each role with the set of abstract objects accessible to the said role, said association requiring neither redundant storage and maintenance of permissions nor exhaustive system searches.

2. The method of claim 1, further comprising:

defining and managing the abstract permissions of a role on abstract objects; and

finding, retrieving, and displaying abstract permissions of a role on abstract objects; and

adding an abstract object to the set of abstract objects associated with a role whenever said abstract object becomes accessible to said role; and

deleting an abstract object from the set of abstract objects associated with a role whenever said abstract object becomes inaccessible to said role.

3. The method of claim 2, further comprising:

creating, finding, retrieving, displaying, and deleting instances of a role on a host computer or set of host computers, using group nesting and a directed acyclic graph of role-membership inheritance; and

creating finding, retrieving, displaying, and deleting object instances of abstract objects on a host computer or set of host computers; and

registering objects as instances of abstract objects on a host computer or set of host computers; and

deriving permissions of a role instance on object instances from the abstract permissions of said role on said abstract objects; and

registering permissions on objects as instances of abstract permissions on abstract objects on a host computer or set of host computers; and

finding, retrieving, and displaying the permissions derived from abstract permissions defined on abstract objects.

4. The method of claim 3, further comprising the steps of:

creating an instance of a RBAC user on a set of host computers, said user instance being called global with respect to said set of host computers; and

creating an instance of a RBAC user on a host computer, said user instance being called local with respect to said host computer, unless said host computer is used to control a set of host computers, in which case the instance is called global with respect to said set of host computers; and

creating a role instance on a set of host computers, said role instance being called global with respect to said set of host computers; and

creating a role instance on a host computer, said role instance being called local with respect to said host computer, unless said host computer is used to control a set of host computers, in which case one can select whether the instance will be local with respect to said host computer, or global with respect to said set of host computers; and

including a local user instance in a local role instance, if said user is assigned to said role, and both said instances were derived on the same host computer; and

including a global user instance in a local role instance, if said user is assigned to said role, and said local role instance was derived on a host computer included in the set of host computers used to derive said global user instance; and

including the global user instance in a global role instance, if said user is assigned to said role, and both said instances were derived on the same set of host computers; and

including the members of a local instance of a first role in a local instance of a second role, if the second role inherits the membership of the first role, and both said instances were derived on the same host computer; and

including the global instance of a first role as a member of a local instance of a second role, if the second role inherits the membership of the first role, and said local instance was derived on a host computer included in the set of host computers used to derive said global instance; and

including the members of a global instance of a first role in a global instance of a second role, if the second role inherits the membership of the first role, and both said instances were derived on the same set of host computers.

5. The method of claim 3, further comprising :

computing, displaying, reviewing, and listing the permissions of any role to abstract objects; and

computing, displaying, reviewing, and listing the permissions of any role to object instances; and

computing, displaying, reviewing, and listing the permissions of any role instance to object instances.

6. The method of claim 5, further comprising:

determining whether two or more roles share permissions on any abstract objects; and

determining whether two or more roles share permissions on any object instances; and

determining whether two or more role instances share permissions on any object instances; and

implementing and testing any policy that is satisfied by the determination of whether two or more roles share permissions to abstract objects; and

implementing and testing any policy that is satisfied by the determination of whether two or more roles share permissions to object instances; and

implementing and testing any policy that is satisfied by the determination of whether two or more role instances share permissions to object instances.

7. The method of claim 6, further comprising:

implementing and testing generalized separation-of-duty policies; and

implementing and testing operational separation-of-duty policies.

8. The method of claim 3, further comprising:

automatic distribution of permissions on object instances to role instances whenever new permission-inheritance relations are established among roles; and

automatic distribution of permissions on object instances to role instances whenever new roles are added to the directed acyclic graph; and

automatic distribution of permissions on object instances to role instances whenever a new role instance is created for a role on a host computer or set of host computers; and

automatic distribution of permissions on object instances to role instances whenever a new object instance is created for an abstract object on a host computer or set of host computers; and

automatic distribution of permissions on object instances to role instances whenever a new permission is granted to a role.

9. The method of claim 3, further comprising:

automatic revocation and recalculation of permissions on object instances for role instances whenever permission-inheritance relations among roles are removed; and

automatic revocation and recalculation of permissions on object instances for role instances whenever roles are removed; and

automatic revocation and recalculation of permissions on object instances for roles instances whenever an abstract object is removed; and

automatic revocation and recalculation of permissions on object instances for role instances whenever a permission is revoked from a role.

10. The method of claim 3, further comprising:

scaleable, automatic, distribution, revocation, and recalculation of permissions of role instances to object instances that support efficient access authorization.

11. The method of claim 10, further comprising:

adding a new permission-inheritance arc to the directed acyclic graph between a first role called inheritor role and a second role called the inherited role whereby the inheritor and all its ascendant roles inherit all the permissions of the inherited role and its descendant roles in the directed acyclic graph; and

automatically selecting the roles that do not have instances on a host computer or set of host computers from the set comprises the said inherited role and its descendants in the directed acyclic graph; and

automatically computing a set of permissions by mapping the abstract permissions of said selected roles on all abstract objects that do have instances on said host computer or set of host computers; and

automatically granting said computed permissions to the instance of each first encountered role instantiated on said host computer or set of host computers by traversing the directed acyclic graph in the direction opposite to that of the inheritance arcs on any path starting from the inheritor role.

12. The method of claim 11, further comprising:

removing a permission-inheritance arc from the directed acyclic graph between a first role called inheritor role and a second role called the inherited role; and

automatically recalculating permissions and granting said permissions to the instance of each first encountered role instantiated on a host computer or set of host computers, by traversing the directed acyclic graph in the direction opposite to that of the inheritance arcs on any path starting from the inheritor role.

13. The method of claim 11, further comprising:

revoking an abstract permission to an abstract object from a role where said abstract object has an instance on a host computer or set of host computers; and

automatically updating the association between the said role and the set of accessible abstract objects; and

automatically recalculating permissions and granting said permissions to the instance of each first encountered role instantiated on a host computer or set of host computers, by traversing the directed acyclic graph in the direction opposite to that of the inheritance arcs on any path starting from the said role.

14. The method of claim 11, further comprising

deleting a role from the directed acyclic graph, further comprising:

selecting a role for deletion from the directed acyclic graph;

automatically removing the said role from the access control lists of all abstract objects accessible to said role; and

automatically deleting the association between said role and all abstract objects accessible to said role; and

automatically recalculating permissions and granting said permissions to the instance of each first encountered role instantiated on a host computer or set of host computers, by traversing the directed acyclic graph in the direction opposite to that of the inheritance arcs on any path starting from the any direct ascendant of the selected; and

automatically deleting all instances of the selected; and

automatically deleting the selected role from the directed acyclic graph.

15. The method of claim 10, further comprising:

creating an instance of a role on a host computer or set of host computers; and

automatically selecting the roles that did not have instances on said host computer or set of host computers prior to the creation of said role instance, wherein the selection is performed from said role and its descendant roles in the directed acyclic graph; and

automatically computing a set of permissions by mapping the abstract permissions of said selected roles on all abstract objects that do have instances on said host computer or set of host computers; and

automatically granting said computed permissions to said role instance just created.

16. The method of claim 10, further comprising:

creating an instance of a user on a host computer or set of host computers; and

automatically selecting the roles that did not have instances on said host computer or set of host computers prior to the creation of said user instance, wherein the selection is performed from said user and its descendant roles in the directed acyclic graph; and

automatically computing a set of permissions by mapping the abstract permissions of said selected roles on all abstract objects that do have instances on said host computer or set of host computers; and

automatically granting said computed permissions to said user instance just created.

17. The method of claim 10, further comprising:

granting a role an abstract permission to an abstract object that has an instance on a host computer or set of host computers and automatically causing the said role's ascendant roles and users to inherit the said abstract permission; and

automatically updating the association between the said role and the set of accessible abstract objects; and

automatically mapping the said abstract permission of said role on said abstract object to a set of permissions for the object instance; and

automatically granting said set of permissions to the instance of each first encountered role instantiated on said host computer or set of host computers by traversing the directed acyclic graph in the direction opposite to that of the inheritance arcs on any path starting from the role being granted the abstract permission.

18. The method of claim 10, further comprising:

instantiating an abstract object on a host computer or set of host computers; and

automatically reading the access control list of the abstract object and computing the set of roles that have abstract permissions to the said abstract object; and

for each role in the said set, automatically mapping the abstract permissions of said role on said abstract object to a set of permissions for the object instance; and

automatically granting said set of permissions to the instance of each first encountered role instantiated on said host computer or set of host computers by traversing the directed acyclic graph in the direction opposite to that of the inheritance arcs on any path starting from said role.

19. The method of claim 10, further comprising

deleting an abstract object, including the steps:

automatically finding and deleting all instances of said abstract object and their access control lists; and

automatically reading the access control list of said abstract object and, for each role found in the said access control list, removing the said abstract object from the association between said role and its set of accessible abstract objects; and

automatically deleting the said abstract object and its access control list.

20. The method of claim 10, further comprising:

deriving a directed acyclic graph of roles representing both membership and permission inheritance, abstract objects, and abstract permissions, from the user account, group, and access control list and permission structures of extant operating systems; and

performing the incremental transition from an extant permission management system to automatic permission management in RBAC.

21. The method of claim 20, further comprising:

deriving membership-inheritance and permission-inheritance relationships among the existing user accounts and groups; and

creating roles and assigning selected user accounts and groups to said roles; and

deriving membership-inheritance and permission-inheritance relationships among said roles and obtaining a directed acyclic graph for each type of inheritance relationship; and

transforming the said directed acyclic graphs into a single directed acyclic graph of membership inheritance that preserves the permission of the user accounts defined by permission inheritance.

22. A computer program product containing computer readable code for causing a machine to perform the following method steps:

automatic distribution, review and revocation of user and group permissions to objects through management of role permissions to abstract objects in a computing environment comprises a role-based access control system that includes a directed acyclic graph representing role-membership inheritance relationships and a directed acyclic graph representing role-permission inheritance relationships;

association of each role with the set of abstract objects accessible to the said role, said association requiring neither redundant storage and maintenance of permissions nor exhaustive system searches.

23. A program product as defined in claim 22, further comprising code for performing the following method steps:

defining and managing the abstract permissions of a role on abstract objects;

finding, retrieving, and displaying abstract permissions of a role on abstract objects;

adding an abstract object to the set of abstract objects associated with a role whenever said abstract object becomes accessible to said role; and

deleting an abstract object from the set of abstract objects associated with a role whenever said abstract object becomes inaccessible to said role.

24. A program product as defined in claim 23, further comprising code for performing the following method steps:

creating, finding, retrieving, displaying, and deleting instances of a role on a host computer or set of host computers, using group nesting and a directed acyclic graph of role-membership inheritance;

creating finding, retrieving, displaying, and deleting object instances of abstract objects on a host computer or set of host computers;

registering objects as instances of abstract objects on a host computer or set of host computers;

deriving permissions of a role instance on object instances from the abstract permissions of said role on said abstract objects;

registering permissions on objects as instances of abstract permissions on abstract objects on a host computer or set of host computers; and

finding, retrieving, and displaying the permissions derived from abstract permissions defined on abstract objects.

25. A program product as defined in claim 24, further comprising code for performing the following method steps:

creating an instance of a RBAC user on a set of host computers, said user instance being called global with respect to said set of host computers;

creating an instance of a RBAC user on a host computer, said user instance being called local with respect to said host computer, unless said host computer is used to control a set of host computers, in which case the instance is called global with respect to said set of host computers;

creating a role instance on a set of host computers, said role instance being called global with respect to said set of host computers;

creating a role instance on a host computer, said role instance being called local with respect to said host computer, unless said host computer is used to control a set of host computers, in which case one can select whether the instance will be local with respect to said host computer, or global with respect to said set of host computers;

including a local user instance in a local role instance, if said user is assigned to said role, and both said instances were derived on the same host computer;

including a global user instance in a local role instance, if said user is assigned to said role, and said local role instance was derived on a host computer included in the set of host computers used to derive said global user instance;

including the global user instance in a global role instance, if said user is assigned to said role, and both said instances were derived on the same set of host computers;

including the members of a local instance of a first role in a local instance of a second role, if the second role inherits the membership of the first role, and both said instances were derived on the same host computer;

including the global instance of a first role as a member of a local instance of a second role, if the second role inherits the membership of the first role, and said local instance was derived on a host computer included in the set of host computers used to derive said global instance; and

including the members of a global instance of a first role in a global instance of a second role, if the second role inherits the membership of the first role, and both said instances were derived on the same set of host computers.

26. A program product as defined in claim 24, further comprising code for performing the following method steps:

computing, displaying, reviewing, and listing the permissions of any role to abstract objects; and

computing, displaying, reviewing, and listing the permissions of any role to object instances; and

computing, displaying, reviewing, and listing the permissions of any role instance to object instances.

27. A program product as defined in claim 26, further comprising code for performing the following method steps:

determining whether two or more roles share permissions on any abstract objects; and

determining whether two or more roles share permissions on any object instances; and

determining whether two or more role instances share permissions on any object instances; and

implementing and testing any policy that is satisfied by the determination of whether two or more roles share permissions to abstract objects; and

implementing and testing any policy that is satisfied by the determination of whether two or more roles share permissions to object instances; and

implementing and testing any policy that is satisfied by the determination of whether two or more role instances share permissions to object instances.

28. A program product as defined in claim 27, further comprising code for performing the following method steps:

implementing and testing generalized separation-of-duty policies; and

implementing and testing operational separation-of-duty policies.

29. A program product as defined in claim 24, further comprising code for performing the following method steps:

automatic distribution of permissions on object instances to role instances whenever new permission-inheritance relations are established among roles; and

automatic distribution of permissions on object instances to role instances whenever new roles are added to the directed acyclic graph; and

automatic distribution of permissions on object instances to role instances whenever a new role instance is created for a role on a host computer or set of host computers; and

automatic distribution of permissions on object instances to role instances whenever a new object instance is created for an abstract object on a host computer or set of host computers; and

for automatic distribution of permissions on object instances to role instances whenever a new permission is granted to a role.

30. A program product as defined in claim 24, further comprising code for performing the method steps of:

automatic revocation and recalculation of permissions on object instances for role instances whenever permission-inheritance relations among roles are removed; and

automatic revocation and recalculation of permissions on object instances for role instances whenever roles are removed; and

automatic revocation and recalculation of permissions on object instances for roles instances whenever an abstract object is removed; and

automatic revocation and recalculation of permissions on object instances for role instances whenever a permission is revoked from a role.

31. A program product as defined in claim 24, further comprising code for performing the method step of:

scaleable, automatic, distribution, revocation, and recalculation of permissions of role instances to object instances that support efficient access authorization.

32. A program product as defined in claim 31, further comprising code for performing the method steps of:

adding a new permission-inheritance arc to the directed acyclic graph between a first role called inheritor role and a second role called the inherited role whereby the inheritor and all its ascendant roles inherit all the permissions of the inherited role and its descendant roles in the directed acyclic graph; and

automatically selecting the roles that do not have instances on a host computer or set of host computers from the set comprises the said inherited role and its descendants in the directed acyclic graph; and

automatically computing a set of permissions by mapping the abstract permissions of said selected roles on all abstract objects that do have instances on said host computer or set of host computers; and

automatically granting said computed permissions to the instance of each first encountered role instantiated on said host computer or set of host computers by traversing the directed acyclic graph in the direction opposite to that of the inheritance arcs on any path starting from the inheritor role.

33. A program product as defined in claim 32, further comprising code for performing the method steps of:

removing a permission-inheritance arc from the directed acyclic graph between a first role called inheritor role and a second role called the inherited role; and

automatically recalculating permissions and granting said permissions to the instance of each first encountered role instantiated on a host computer or set of host computers, by traversing the directed acyclic graph in the direction opposite to that of the inheritance arcs on any path starting from the inheritor role.

34. A program product as defined in claim 32, further comprising code for performing the method steps of:

revoking an abstract permission to an abstract object from a role where said abstract object has an instance on a host computer or set of host computers; and

automatically updating the association between the said role and the set of accessible abstract objects; and

automatically recalculating permissions and granting said permissions to the instance of each first encountered role instantiated on a host computer or set of host computers, by traversing the directed acyclic graph in the direction opposite to that of the inheritance arcs on any path starting from the said role.

35. A program product as defined in claim 32, further comprising code for performing the method steps of

deleting a role from the directed acyclic graph, further comprising:

selecting a role for deletion from the directed acyclic graph;

automatically removing the said role from the access control lists of all abstract objects accessible to said role; and

automatically deleting the association between said role and all abstract objects accessible to said role; and

automatically recalculating permissions and granting said permissions to the instance of each first encountered role instantiated on a host computer or set of host computers, by traversing the directed acyclic graph in the direction opposite to that of the inheritance arcs on any path starting from the any direct ascendant of the selected; and

automatically deleting all instances of the selected; and

automatically deleting the selected role from the directed acyclic graph.

36. A program product as defined in claim 31, further comprising code for performing the method steps of:

creating an instance of a role on a host computer or set of host computers; and

automatically selecting the roles that did not have instances on said host computer or set of host computers prior to the creation of said role instance, wherein the selection is performed from said role and its descendant roles in the directed acyclic graph; and

automatically computing a set of permissions by mapping the abstract permissions of said selected roles on all abstract objects that do have instances on said host computer or set of host computers; and

automatically granting said computed permissions to said role instance just created.

37. A program product as defined in claim 31, further comprising code for performing the method steps of:

creating an instance of a user on a host computer or set of host computers; and

automatically selecting the roles that did not have instances on said host computer or set of host computers prior to the creation of said user instance, wherein the selection is performed from said user and its descendant roles in the directed acyclic graph; and

automatically computing a set of permissions by mapping the abstract permissions of said selected roles on all abstract objects that do have instances on said host computer or set of host computers; and

automatically granting said computed permissions to said user instance just created.

38. A program product as defined in claim 31, further comprising code for performing the method steps of:

granting a role an abstract permission to an abstract object that has an instance on a host computer or set of host computers and automatically causing the said role's ascendant roles and users to inherit the said abstract permission; and

automatically updating the association between the said role and the set of accessible abstract objects; and

automatically mapping the said abstract permission of said role on said abstract object to a set of permissions for the object instance; and

automatically granting said set of permissions to the instance of each first encountered role instantiated on said host computer or set of host computers by traversing the directed acyclic graph in the direction opposite to that of the inheritance arcs on any path starting from the role being granted the abstract permission.

39. A program product as defined in claim 31, further comprising code for performing the method steps of:

instantiating an abstract object on a host computer or set of host computers; and

automatically reading the access control list of the abstract object and computing the set of roles that have abstract permissions to the said abstract object; and

for each role in the said set, automatically mapping the abstract permissions of said role on said abstract object to a set of permissions for the object instance; and automatically granting said set of permissions to the instance of each first encountered role instantiated on said host computer or set of host computers by traversing the directed acyclic graph in the direction opposite to that of the inheritance arcs on any path starting from said role.

40. A program product as defined in claim 31, further comprising code for performing the method steps of

deleting an abstract object, further comprising code for:

automatically finding and deleting all instances of said abstract object and their access control lists; and

automatically reading the access control list of said abstract object and, for each role found in the said access control list, removing the said abstract object from the association between said role and its set of accessible abstract objects; and

automatically deleting the said abstract object and its access control list.

41. A program product as defined in claim 31, further comprising code for performing the method steps of:

deriving a directed acyclic graph of roles representing both membership and permission inheritance, abstract objects, and abstract permissions, from the user account, group, and access control list and permission structures of extant operating systems; and

performing the incremental transition from an extant permission management system to automatic permission management in RBAC.

42. A program product as defined in claim 31, further comprising code for performing the method steps of:

deriving membership-inheritance and permission-inheritance relationships among the existing user accounts and groups; and

creating roles and assigning selected user accounts and groups to said roles; and

deriving membership-inheritance and permission-inheritance relationships among said roles and obtaining a directed acyclic graph for each type of inheritance relationship; and

transforming the said directed acyclic graphs into a single directed acyclic graph of membership inheritance that preserves the permission of the user accounts defined by permission inheritance.